CS422 - Final Exam

Time: 3 hours
Total: 100 points
You can use books, lecture notes, tea, coffee, hammers, axes, etc., but *no laptops*!

Problem 1. (15 points)
Consider programs of the following form, written in an uncurried `FUN`-like language supporting the various parameter passing styles discussed in class:

```
let f(P x, P y) = x + y
and g(Q x, Q y) = {
                      x := x + y ;
                      y := x - y ;
                      x := x - y ;
                      x + y
                    }
and x = 5 and y = 7
in f(g(x,y), g(y,y))
```

Here `P` and `Q` stay for parameter passing styles. What values do these programs evaluate to, when `P` is any of call-by-value, call-by-name, or call-by-need, and when `Q` is any of call-by-reference or call-by-value-result? For each of the six situations, give also a brief explanation of how you obtained the result.

Problem 2. (15 points)
Parallel assignments, typically written `X1, X2, ..., Xn := E1, E2, ..., En`, can be a useful feature of a programming language, because they allow one to write more compact, abstract and intuitive code. A parallel assignment first evaluates the expressions in its right-hand-side and then assigns the obtained values to the names listed in its left-hand-side, in the corresponding order. For example, `x,y := y,x` swaps the values of `x` and `y` without a need for a temporary variable. Suppose that you, as a language designer, want to add such parallel assignments to your programming language, in this case `FUN`. This problem has two parts:

1. Show that the parallel assignment feature is just "syntactic sugar", in the sense that it can be automatically translated into an equivalent expression using only the already existing `FUN` language features;

2. Give the parallel assignment a direct continuation-based semantics.

For this problem you can assume any of the continuation-based semantics of `FUN` that we discussed in class, together with all its provided infrastructure. In short, you do not need to redefine anything that was already defined in class.

Problem 3. (15 points)
Suppose that you execute the object oriented program below:

```
class A {
  field value;
  method initialize(v) { value := v }
  method m() { value + 10 }
}
class B inherits A {
  method m() { super m() }
}
class C inherits B {
  field obj;
  method void initialize(v,o) {
    super initialize(v);
    obj := o
  }
  method m() { value + send obj m() }
}
class D inherits B {
  method initialize(v) { super initialize(v) }
  method m() { value * 2 }
}
main
  let b = ...
  in let o = if b then new C(5, new C(10, new D(15))) else new D(20)
     in send o m()
```

What value will be returned under dynamic method invocation when b is true? When b is false? What value will be returned under static method invocation when b is true? When b is false? Assume we are in a typed environment, even without typing annotations, so you should take the types of expressions into account. Justify your answers briefly.

Problem 4. (10 points)
Type the expression bellow, following the pre-typing and the unification-based typing methods discussed in class for the curried version of `FUN`:

```
let h x y z u = if (x u) eq (y u) then z u else u
in h (fun x -> x)
```

You do not need to do it at the last detail, but make sure that you include enough detail to convince us that you understand the discussed type inference technique well.

Problem 5. (15 points)
Transform the functional program below into a continuation passing style equivalent program, by applying the CPS procedure discussed in class. Like in the problem above, go into as much detail as needed in order to show that you understand the CPS procedure well.

```
let rec
    perm(n) =
      if n eq 0 then [[]] else insert(n, perm(n - 1))
and insert(n,l) =
      if null?(l) then []
      else append(interleave(n,car(l)), insert(n,cdr(l)))
and interleave(n,l) =
      if null?(l) then [[n]]
      else cons(cons(n,l), mycons(car(l), interleave(n,cdr(l))))
and mycons(x,l) =
      if null?(l) then [] else cons(cons(x,car(l)), mycons(x,cdr(l)))
and append(u,v) =
      if null?(u) then v else cons(car(u), append(cdr(u), v))
in perm(3)
```

Problem 6. (15 points)
Replace the question mark "?" in the partial correctness assertion below by the corresponding expression, and then prove it using the Hoare rules:

$\{N > 0\}$

```
S = 0;  P = 0;  T = 1 ;
while (P < N) (
  S = S + T; P = P + 1; T = 2 * T
)
```

$\{S = ?\}$

Problem 7. (15 points)
Consider the following program in `CONCURRENT-FUN`:

```
let c = 0
in {
     while (c leq 1) {
        spawn(c := c * c) ;
        c := c + 1
     } ;
     c
   }
```

Can this program evaluate to the integer 16? What is the minimum number of threads that may run concurrently in some execution at any given moment? But the maximum? What is the smallest number that can be returned by this program? But the maximum? Add appropriate thread synchronization to this program so that it becomes free of data-races. What are the possible values returned by the data-race-free program?